PARTICLE SWARM OPTIMIZATION ALGORITHM: VARIANTS AND COMPARISONS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Sowjanya Mattaparthi

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

October 2015

Fargo, North Dakota

North Dakota State University
Graduate School

Title

PARTICLE SWARM OPTIMIZATION ALGORITHM:
VARIANTS AND COMPARISONS

By

Sowjanya Mattaparthi

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Saeed Salem

Dr. María de los Ángeles Alfonseca-Cubero

Approved:

10/29/2015
Date

Dr. Brian M. Slator
Department Chair

ABSTRACT

Since the introduction of Particle Swarm optimization by Dr. Eberhart and Dr. Kennedy, there have been many variations of the algorithm proposed by many researchers and various applications presented using the algorithm. In this paper, we applied variants of Particle swarm optimization on various benchmark functions in multiple dimensions, using the computational procedure to find the optimal solutions for those functions. We ran the variants of the algorithm 51 times on each of the 17-benchmark functions and computed the average, variance and standard deviation for 10, 30, and 50 dimensions. Using the results, we found the suitable variants of the algorithm for the benchmark functions by considering the minimum optimal solution produced by each variant.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Simone Ludwig for the continuous support and valuable guidance she has provided throughout my research. Her research experience and knowledge inspired and motivated me to work harder for my Master's paper. I would also like to thank Dr. Alfonseca and Dr. Salem for accepting to be in my committee, and providing their valuable inputs and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

1. INTRODUCTION

"Particle swarm optimization (PSO) is a stochastic optimization approach, modeled on the social behavior of bird flocks" [1]. Particle Swarm optimization is derived from a broader area called Swarm Intelligence, which in turn is a subtopic of Computational Intelligence.

Andries P. Engelbrecht in his book, defined Computational Intelligence as "the study of adaptive mechanisms to enable or facilitate intelligent behavior in complex and changing environments" [1]. He also mentioned that Swarm Intelligence is one of the technological paradigms of Computational Intelligence. Swarm intelligence comes from swarming behaviors of groups of organisms [2]. Other areas of Computational intelligence include Artificial Immune Systems inspired by natural immune systems, Artificial Neural Networks inspired by biological neural networks, and Evolutionary computation inspired by evolution of organisms that has been going on from thousands of years on this earth.

Individuals in a swarm may have strengths and skillsets of varied magnitude, but they collectively work together to perform great work that they would not have done if each worked on their own without any coordination [3].

Swarm intelligence is an innovative distributed artificial intelligence paradigm for solving optimization problems that originally took its inspiration from the collective behavior of social insects such as ants, termites, bees, and wasps, as well as from other animal societies such as flocks of birds or schools of fish [4]. It is amazing how insects organize themselves as a group in accomplishing tasks such as building nests, searching for food, gathering food, defending themselves against attacks, etc. This is all done without any sophisticated communication devices.

Douglas H. Chadwick in his article “Sisterhood of Weavers” explained how the weaver ants work together in building their nests, and how they communicate [5]. Each weaver ant colony inhabits from half a dozen to more than a hundred nests at any given time, forming a metropolis of boroughs and suburbs connected by busy commuter routes. A hierarchy of workers and soldiers maintains and defends this territory, which spreads from treetops to the forest floor, staying in sync through constant communication [5]. They touch each other with mouths, forelegs, or antennae. They lay down scents with different glands to send different messages. They release more pheromones into the air to broadcast signals quickly and widely. They even display symbolic behavior: To warn of an approaching enemy, for instance, they jerk their bodies in a kind of ritualized fight” [5].

Andries P. Engelbrecht [1] gave an excellent analogy of Swarm in his book. His example mentions a team of people searching for a treasure. All of them use a metal detector to continuously detect any signal from the device. The group communicates with each other to share the information on the strength of the signals they get, thereby using that information to find the proximity to the treasure.

In this paper, we implemented the computational procedure for the Particle swarm optimization and applied various variants of it on various benchmark functions in multiple dimensions to find the optimal solutions for those functions. After various runs of the algorithm to get the average, variance and the standard deviation from the results, we found out which variant of the algorithm gives the best (minimal optimal solution) for the various benchmark functions.

Particle Swarm Optimization and Ant colony optimization are closely related and one of the most researched topics today. These two models were developed based on the study of the organisms. Simulation studies of the graceful, but unpredictable, choreography of bird flocks led

to the design of the particle swarm optimization algorithm, and studies of the foraging behavior of ants resulted in ant colony optimization algorithms [1].

These two models helped solve many problems and applied in multiple areas. For example, in the paper "A hybrid particle swarm optimization for dynamic facility layout problem" [6], the authors proposed a hybrid particle swarm optimization (HPSO) algorithm to find near optimal solutions of dynamic facility layout problem. The dynamic facility layout problem (DFLP) aims to minimize the sum of handling and re-layout costs by devising an individual layout for each distinctive production period [6].

And, Ant Colony Optimization helped in solving Travelling salesman problem. Given a collection of cities and the cost of travel between each pair of them, the **traveling salesman problem**, or **TSP** for short, is to find the cheapest way of visiting all of the cities and returning to your starting point [7].

Marco Dorigo and Luca Maria Gambardella described an artificial ant colony capable of solving the traveling salesman problem. Ants of the artificial colony are able to generate successively shorter feasible tours by using information accumulated in the form of a pheromone trail deposited on the edges of the TSP graph [8]. Computer simulations demonstrate that the artificial ant colony is capable of generating good solutions to both symmetric and asymmetric instances of the TSP [8].

2. PARTICLE SWARM OPTIMIZATION ALGORITHM

2.1. Basic PSO

In Basic PSO, particles are "flown" through a multidimensional search space, where the position of each particle is adjusted according to its own experience and that of its neighbors [1].

Let $x_i(t)$ denote the position of particle i in the search space at time step t; unless otherwise stated, t denotes discrete time steps. The position of the particle is changed by adding a velocity, $v_i(t)$, to the current position [1], i.e.,

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (1)$$

It is the velocity vector that drives the optimization process, and reflects both the experiential knowledge of the particle and socially exchanged information from the particle's neighborhood [1]. This is one of the best ways of reaching optimal solution in a collaborative scenario.

The basic PSO is influenced by a number of control parameters, namely the dimension of the problem, number of particles, acceleration coefficients, inertia weight, neighborhood size, number of iterations, and the random values that scale the contribution of the cognitive and social components [1].

One of the other control parameters is swarm size. It is the number of particles in the swarm. Initial diversity of the swarm increases when there are more particles in the swarm. A large swarm allows larger parts of the search space to be covered per iteration. The downside to this is, larger swarm size increases the per iteration computational complexity, and the search degrades to a parallel random search. Larger swarm size may also lead to less iteration to reach a good solution, compared to smaller swarms [1].

Number of iterations is another control parameter. Few iterations may terminate the search prematurely, whereas large number of iterations have the consequence of unnecessary added computational complexity (provided that the number of iterations is the only stopping condition) [1].

Acceleration coefficient is also another control parameter that influences Basic PSO. The acceleration coefficients, c_1 and c_2, together with the random vectors r_1 and r_2, control the stochastic influence of the cognitive and social components on the overall velocity of a particle [1].

2.2. Global Best PSO

For the global best PSO, the neighborhood for each particle is the entire swarm. The social network employed by the $gbest$ PSO reflects the star topology [1]. For the star neighborhood topology, the social component of the particle velocity update reflects information obtained from all the particles in the swarm. In this case, the social information is the best position found by the swarm, referred to as $\hat{y}(t)$ [1].

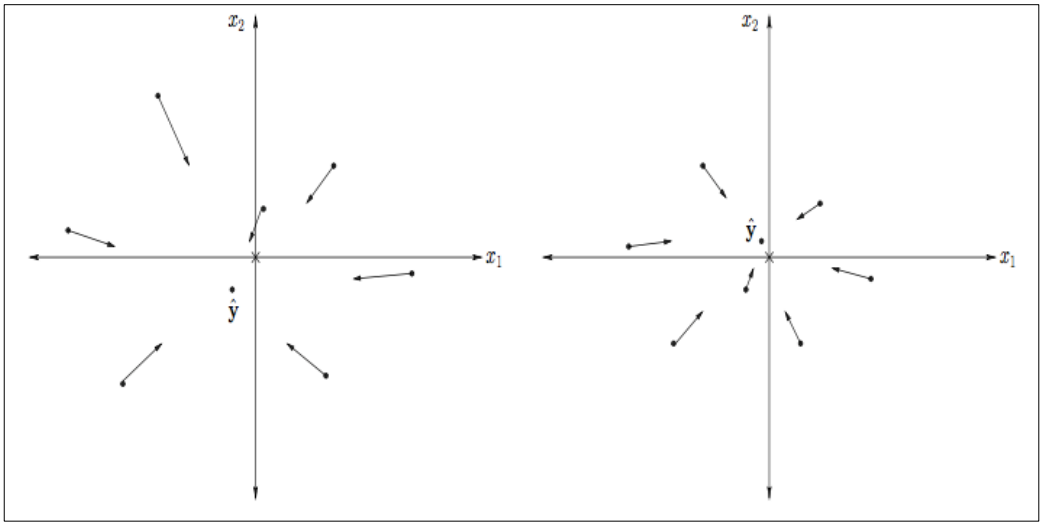

Figure 1. Multi-particle gbest PSO Illustration **[1]**

For $gbest$ PSO, the velocity of particle i is calculated in Equation (2):

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (2)$$

where $v_{ij}(t)$ is the velocity of particle i in dimension $j = 1, \ldots, n_x$ at time step t, $x_{ij}(t)$ is the position of particle i in dimension j at time step t, c_1 and c_2 are positive acceleration constants used to scale the contribution of the cognitive and social components, respectively, and $r_{1j}(t)$, $r_{2j}(t) \sim U(0,1)$ are random values in the range [0,1], sampled from a uniform distribution [1]. These random values introduce a stochastic element to the algorithm [1].

The personal best position, y_i associated with particle i is the best position the particle has visited since the first time step [1]. Considering minimization problems, the personal best position at the next time step, $t+1$ is calculated in Equation (3).

$$y_i(t+1) = \begin{cases} y_i(t) & if\ f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & if\ f(x_i(t+1)) < f(y_i(t)) \end{cases} \quad (3)$$

where $f: \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ is the fitness function [1]. As with evolutionary algorithms, the fitness function measures how close the corresponding solution is to the optimum, i.e. the fitness function quantifies the performance, or quality, of a particle (or solution) [1].

The global best position, $\hat{\boldsymbol{y}}(t)$ at time step t, is defined as in Equation (4):

$$\hat{\boldsymbol{y}}(t) \in \{\boldsymbol{y}_0(t), \ldots, \boldsymbol{y}_{n_s}(t)\} | f(\hat{\boldsymbol{y}}(t)) = min\{f(\boldsymbol{y}_0(t)), \ldots, f(\boldsymbol{y}_{n_s}(t))\} \quad (4)$$

where n_s is the total number of particles in the swarm. It is important to note that the definition in Equation 4 states that $\hat{y}$ is the best position discovered by any of the particles so far, it is usually calculated as the best personal best position [1]. The global best position can also be selected from the particles of the current swarm, in which case:

$$\hat{y}(t) = min\{f(x_0(t)), \ldots, f(x_{n_s}(t))\} \quad (5)$$

2.2.1. Algorithm: gbest PSO [1]

Create and initialize an n_x dimensional swarm;
repeat
for *each particle* $i = 1, \ldots, n_s$ **do**
    //set the personal best position
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
        $\boldsymbol{y}_i = \boldsymbol{x}_i$;
    end
    //set the global best position
    if $f(\boldsymbol{y}_i) < f(\hat{\boldsymbol{y}})$ **then**
        $\hat{\boldsymbol{y}} = \boldsymbol{y}_i$;
    end
end
for *each particle* $i = 1, \ldots, n_s$ do
    update the velocity using equation (2);
    update the position using equation (1);
end
until *stopping condition is true;*

Figure 2. gbest PSO Algorithm **[1]**

2.3. Flowchart of Basic PSO

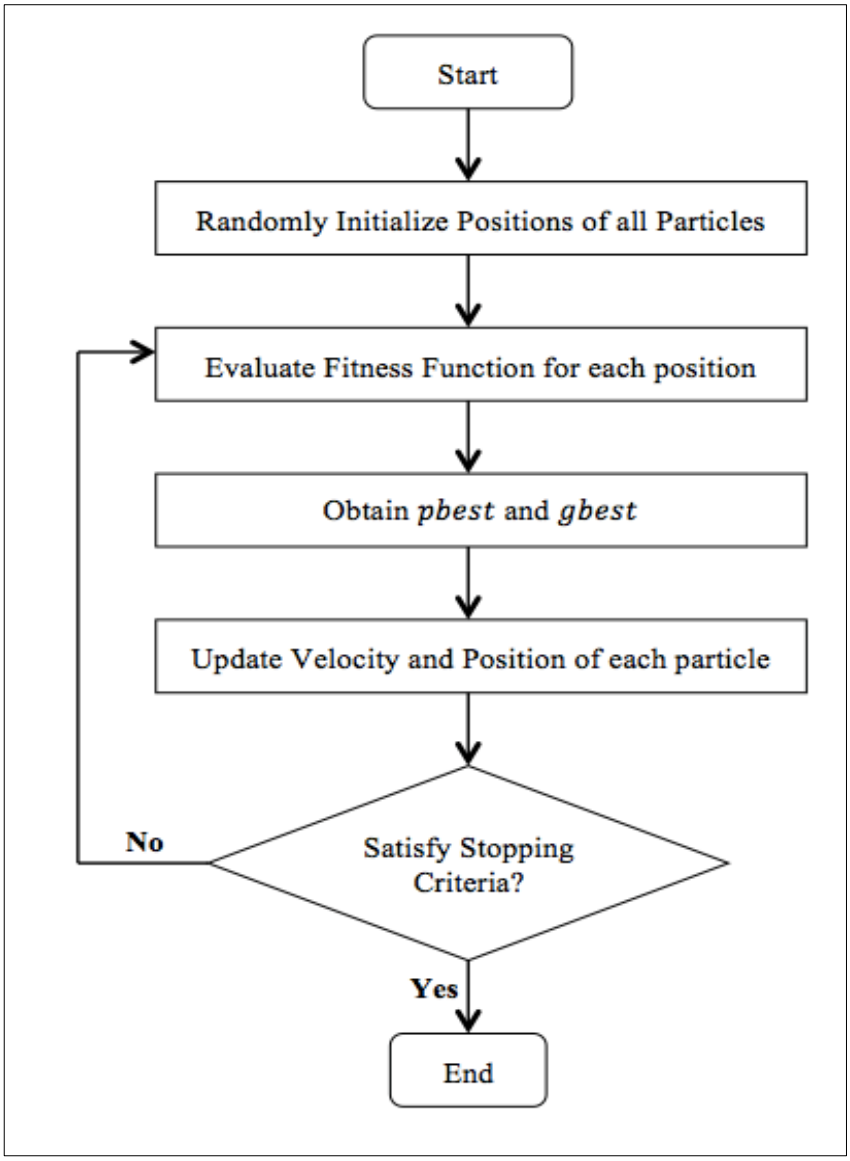

Figure 3. Flowchart of Basic PSO

3. VARIANTS OF PSO

3.1. Basic Particle Swarm Optimization (BPSO)

PSO algorithm starts with a swarm of particles or solutions spread across in a multidimensional search space. Each particle adjusts its position according to its own experience and that of its neighbors. If $x_i(t)$ denotes the position of particle i in the search space at time step t; unless otherwise stated, t denotes discrete time steps [1].

The modified velocity and position of each particle are calculated using Equation (6) and Equation (1),

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (6)$$

where $w = 0.6, c_1 = 2, c_2 = 2$.

3.2. Decreasing Weight Particle Swarm Optimization (DWPSO)

The performance of DWPSO is significantly improved over the original PSO because DWPSO balances out the global and local search abilities of the swarm effectively [9]. In DWPSO, w_{dwpso} is the inertia weight, which linearly decreases from 0.9 to 0.4 through the search process [9].

The modified velocity and position of each particle are calculated using Equation (7) with the linearly decreased weight as in Equation (8),

$$v_{ij}(t+1) = w_{dwpso} v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (7)$$

$$w_{dwpso} = w_{start} - (w_{start} - w_{end}) \times \frac{t}{t_{max}} \quad (8)$$

where w_{start} = 0.9, w_{end} = 0.4, t = current iteration, t_{max}= max no. of iterations.

3.3. Particle Swarm Optimization with Constriction Factor (PSO_C)

Because particle swarm optimization originated from efforts to model social systems, a thorough mathematical foundation for the methodology was not developed at the same time as the algorithm [10]. So, a simplified method of incorporating constriction factor is proposed. Below is Equation (9) with constriction factor incorporated, where K is a function of c_1 and c_2 as in Equation (10).

$$v_{ij}(t+1) = K \times [v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)]] \quad (9)$$

$$K = \frac{2}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|}, where\ \phi = c_1 + c_2, \phi > 4 \quad (10)$$

3.4. Self-Organizing Hierarchical Particle Swarm Optimization with Time Varying Acceleration Coefficients (SPSO_TVAC)

In SPSO_TVAC, if the velocity term reaches zero, particles stagnate due to the lack of momentum to find the global optimal solution. To overcome the limitation, the velocity vector of a particle is reinitialized with a random velocity whenever it stagnates during the search [11].

$$v_{id} = sign(v_{id}) * min\left(abs\left(v_{id}, v_{d,max}\right)\right) \quad (11)$$

where $v_{d,max}$ = 100.

The pseudo code for the SPSO-TVAC method is as follows [11]:

Initialize the population

Update the Velocity

if $v_{id} = 0$

    if rand < 0.5

        v_{id} = rand*v_{dmax}

    else v_{id} = - rand*v_{dmax}

    end if

end if

v_{id}= sign(v_{id})*min(abs(v_{id},v_{dmax}))

Position update using $x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}$

Figure 4. Pseudo Code for SPSO-TVAC

3.5. Time-Varying Particle Swarm Optimization (TVPSO)

With a large c_1 and small c_2 at the beginning, particles are allowed to move around the design space, indicating higher exploration capability [12]. On the other hand, a small value of c_1 and a large value of c_2 signify the convergence to the global best. Therefore, during the exploration process the value of c_2 is decreased from 2.5 to 0.5 and value of c_2 is increased from 0.5 to 2.5 over time [12]. Formally, value of c_1 and c_2 are evaluated as in Equations (12) and (13).

$$c_1 = \left(c_{1f} - c_{1i}\right)\frac{t}{T} + c_{1i} \tag{12}$$

$$c_2 = \left(c_{2f} - c_{2i}\right)\frac{t}{T} + c_{2i} \tag{13}$$

where c_1 is the current acceleration coefficient, c_2 is the current social acceleration coefficient, c_{1i} is the initial cognitive acceleration coefficient = 2.5, c_{1f} is the final cognitive acceleration coefficient = 0.5, c_{2i} is the initial social acceleration coefficient = 0.5, c_{2f} is the final social acceleration coefficient = 2.5, 't' is the current iteration, and 'T' is the total number of iterations [12].

The modified velocity of each particle is calculated using Equation (6) with c_1 and c_2 as in Equations (12) and (13).

3.6. Time-Varying Inertia Weight Particle Swarm Optimization (TVIW_PSO)

In this variant of PSO, value of Inertia weight factor w is decreased linearly from w_{max} to w_{min} as the iteration grows. Suitable selection of the inertia weight provides a balance between global and local exploration abilities, and results in less iteration to find an optimal solution. *w* decreases linearly from 0.9 to 0.4 during iteration [11]. Its value is given in Equation (14).

$$w = (w_{max} - w_{min}) \times \left(\frac{k_{max}-k}{k_{max}}\right) + w_{min} \tag{14}$$

To improve the convergence rate, the constriction factor (*C*) is analyzed by the Eigen value analysis expressed as in Equation (15) [11]:

$$C = \frac{2}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|}, where\ 4.1 \leq \phi \leq 4.2 \tag{15}$$

The convergence characteristics of the system are controlled by ϕ. As ϕ increases, the factor *C* decreases leading to a slower convergence rate because population diversity is reduced [11]. The modified velocity of each particle is calculated with Equation (16):

$$v_{ij}(t+1) = C \times [wv_{ij}(t) + [c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)]]] \tag{16}$$

3.7. Time-Varying Acceleration Coefficients Particle Swarm Optimization (TVAC_PSO)

Even though the TVIW-PSO can determine a better solution in a fast convergence rate, its ability to fine tune the optimal solution is lacking because of diversity at the end of the search. TVAC enhances the global search in the early stage and persuades the particles to converge toward the global optimum at the end of the search. By changing the acceleration coefficients c_1 and c_2 with TVAC, the cognitive component is reduced while the social component increases as the search proceeds [13]. A large cognitive component and a small social component allow the particles to move around the search space instead of moving toward the population best during early stages. By contrast, the particles are allowed to converge to the global optimum in the latter part of optimization. Mathematically, the acceleration coefficients can be formulated as shown in Equations (17) and (18).

$$c_1 = (c_{1f} - c_{1i}\left(\frac{k}{k_{max}}\right) + c_{1i}) \tag{17}$$

$$c_2 = (c_{1f} - c_{2i}\left(\frac{k}{k_{max}}\right) + c_{2i}) \tag{18}$$

The modified velocity of each particle is calculated using Equation (6) with c_1 and c_2 as in Equations (17) and (18).

4. BENCHMARK FUNCTIONS

P. N. Suganthan et al. learnt that there is no standard test suite of problems that the optimization algorithms can be run against. It is hard to prove the efficiency of an algorithm if it is tested with a subset of standard test problems. Since many optimization algorithms such as Particle Swarm Optimization, Evolutionary Programming, Differential evolution, etc. were proposed and will continue to evolve for greater efficiency, a standard test suite is essential. So, in the year 2005, the authors came up with a test suite of 25 benchmark functions [14]. Again in the year 2013, they came up with an improved and additional test functions. In our research, we ran various variants of Particle Swarm Optimization algorithms against 17 CEC'13 Test Functions [15]. Below is the list of Test functions that we ran the PSO algorithms against.

Below is the terminology that is frequently used in the test suite.

$\boldsymbol{D}$: Number of Dimensions

$\boldsymbol{o}$: Shifted global optimum

M1, M2,...,M10: orthogonal (rotation) matrix generated from standard normally distributed entries by Gram-Schmidt ortho-normalization.

Λ^{α}: a diagonal matrix in D dimensions with the i^{th} diagonal element as $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}$, $i = 1,2, \dots, D$.

$T_{asy}^{\beta}: if\ x_i > 0, x_i = x_i^{1+\beta\frac{i-1}{D-1}\sqrt{x_i}}, for\ i = 1,2, \dots D$

$T_{osz}: for\ x_i = sign(x_i)\exp(\hat{x_i} + 0.049(\sin(c_1\hat{x_i}) + \sin(c_2\hat{x_i}))), for\ i = 1\ and\ D$

$$where\ \hat{x_i} = \begin{cases} \log(|x_i|) & if\ \ x_i \neq 0 \\ 0 & otherwise \end{cases}, sign(x_i) = \begin{cases} -1 & if\ x_i > 0 \\ 0 & if\ x_i = 0 \\ 1 & otherwise \end{cases}$$

$$c_1 = \begin{cases} 10 & if\ x_i > 0 \\ 5.5 & otherwise \end{cases}, and\ c_2 = \begin{cases} 7.9 & if\ x_i > 0 \\ 3.1 & otherwise \end{cases}$$

4.1. Sphere Function (F1)

$$f_1(x) = \sum_{i=1}^{D} z_i^2 + f_1^*, z = x - o$$

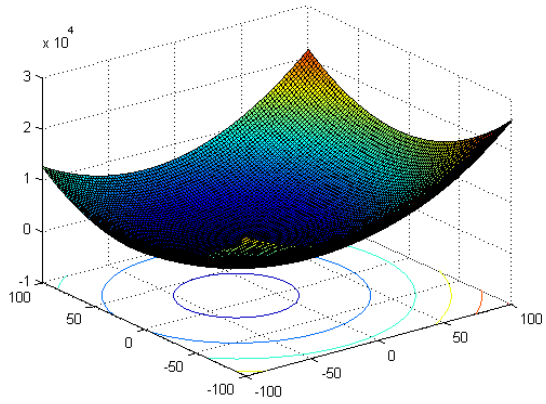

Figure 5. Sphere Function **[15]**

Properties:

- Unimodal
- Separable

4.2. Rotated High Conditioned Elliptic Function (F2)

$$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_2^*, \quad z = T_{osz}(M_1(x-o))$$

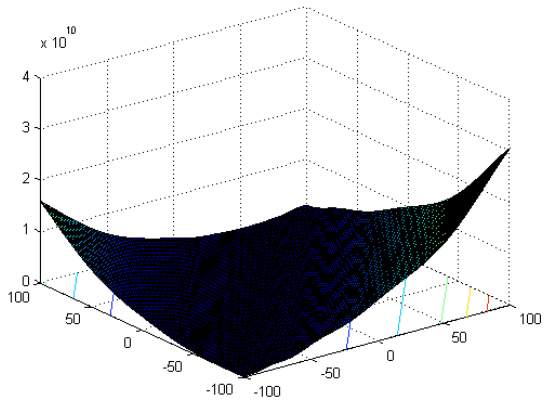

Figure 6. Rotated High Conditioned Elliptic Function **[15]**

Properties:

- Unimodal
- Non-Separable
- Quadratic ill-conditioned
- Smooth local irregularities

4.3. Different Powers Function (F3)

$$f_3(x) = \sqrt{\sum_{i=1}^{D} |z_i|^{2+4\frac{i-1}{D-1}}} + f_3^*, z = x - o$$

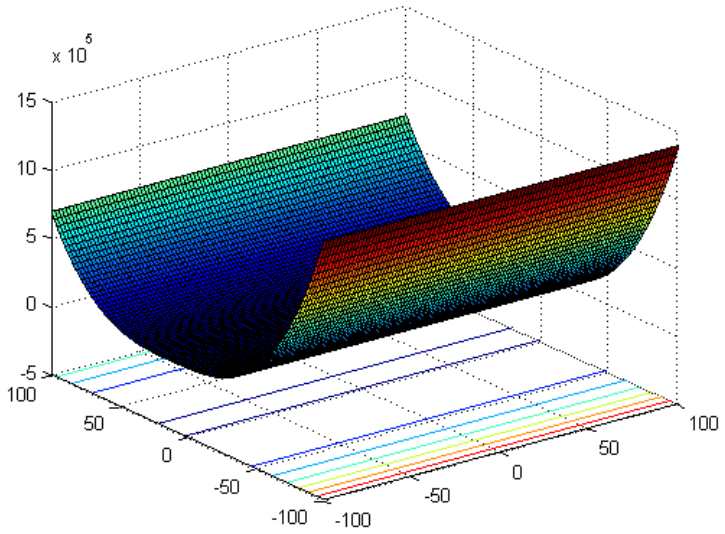

Figure 7. Different Powers Function **[15]**

Properties:

- Unimodal
- Separable
- Sensitivities of the z_i-variables are different.

4.4. Rotated Rosenbrock's Function (F4)

$$f_4(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_4^*, z = M_1\left(\frac{2.048(x-o)}{100}\right) + 1$$

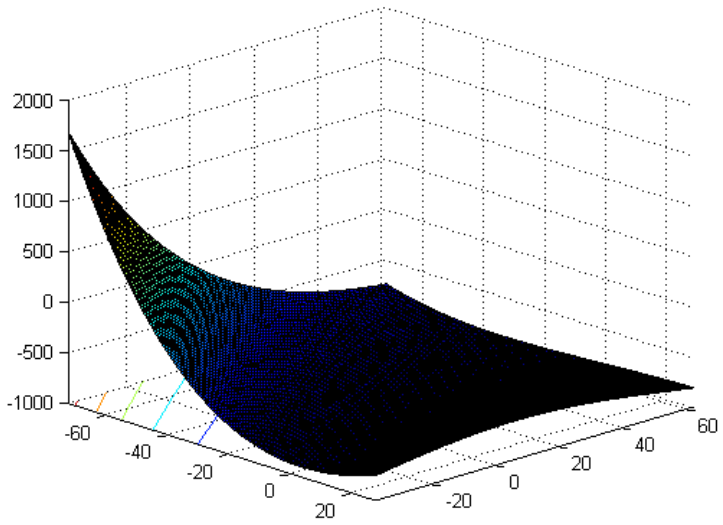

Figure 8. Rotated Rosenbrock's Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Having a very narrow valley from local optimum to global optimum

4.5. Rotated Schaffers Function (F5)

$$f_5(x) = (\frac{1}{D-1}\sum_{i=1}^{D-1}(\sqrt{z_i} + \sqrt{z_i}sin^2(50z_i^{0.2})))^2 + f_5^*$$

$$z_i = \sqrt{y_i^2 + y_{i+1}^2}\, for\ i = 1, \dots, D., y = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(x-o))$$

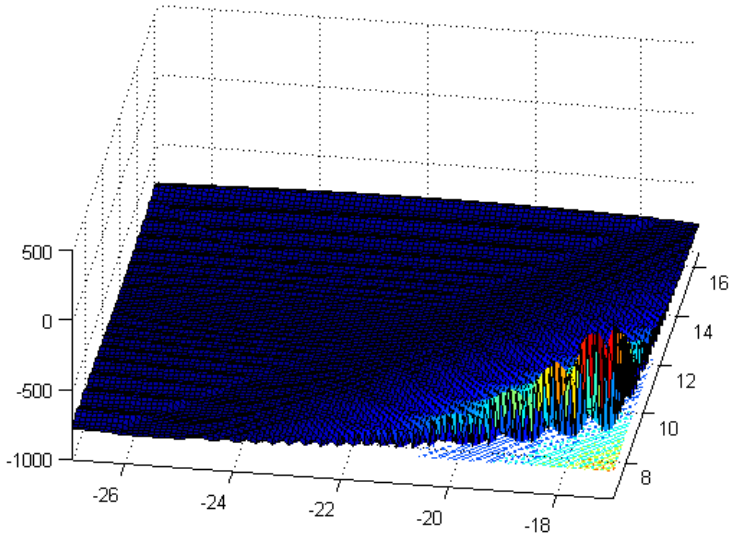

Figure 9. Rotated Schaffers Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Local optima's number in huge

4.6. Rotated Ackley's Function (F6)

$$f_6(x) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i-1}^{D}\cos(2\pi z_i)\right) + 20 + e + f_6^*$$

$$z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(x - o))$$

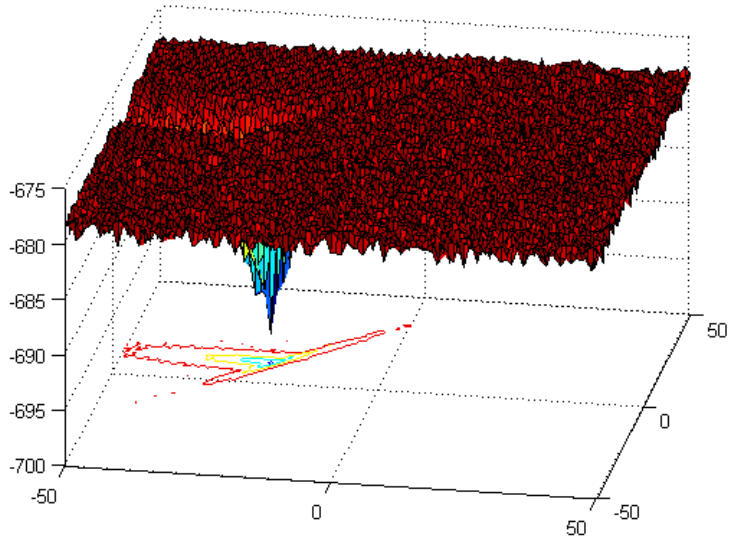

Figure 10. Rotated Ackley's Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical

4.7. Rotated Weierstrass Function (F7)

$$f_7(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{kmax}\left[a^k \cos\left(2\pi b^k (z_i + 0.5)\right)\right]\right) - D\sum_{k=0}^{kmax}\left[a^k \cos(2\pi b^k * 0.5)\right] + f_7^*$$

$$a = 0.5, b = 3, kmax = 20, z = \Lambda^{10} M_2 T_{asy}^{0.5}\left(M_1 \frac{0.5(x-o)}{100}\right)$$

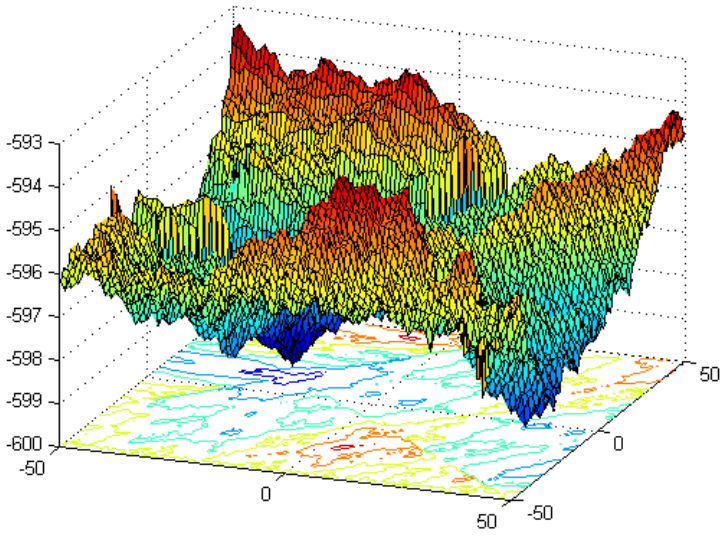

Figure 11. Rotated Weierstrass Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Continuous but differentiable only on a set of points

4.8. Rotated Griewank's Function (F8)

$$f_8(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_8^* \ , \qquad z = \Lambda^{100} M_1 \frac{600(x-o)}{100}$$

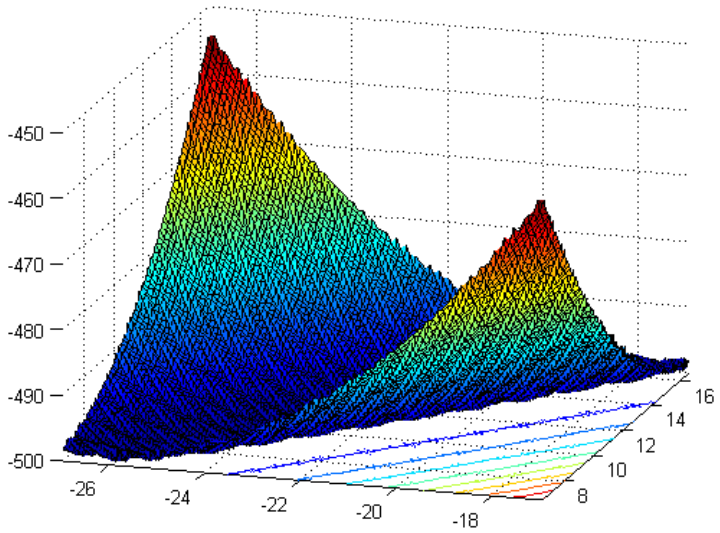

Figure 12. Rotated Griewank's Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical

4.9. Rastrigin's Function (F9)

$$f_9(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_9^*$$

$$z = \Lambda^{10} T_{asy}^{0.2}(T_{osz}(\frac{5.12(x-o)}{100}))$$

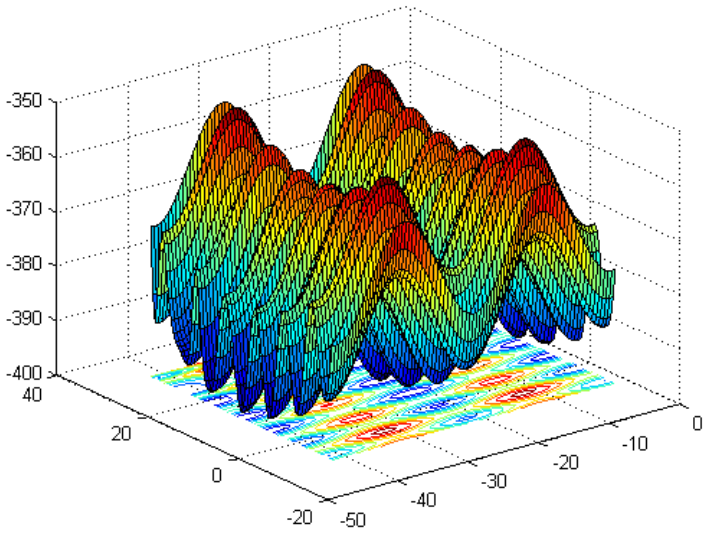

Figure 13. Rastrigin's Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical

4.10. Rotated Rastrigin's Function (F10)

$$f_{10}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{10}^*$$

$$z = M_1\Lambda^{10}M_2T_{asy}^{0.2}(T_{osz}(M_1\frac{5.12(x-o)}{100}))$$

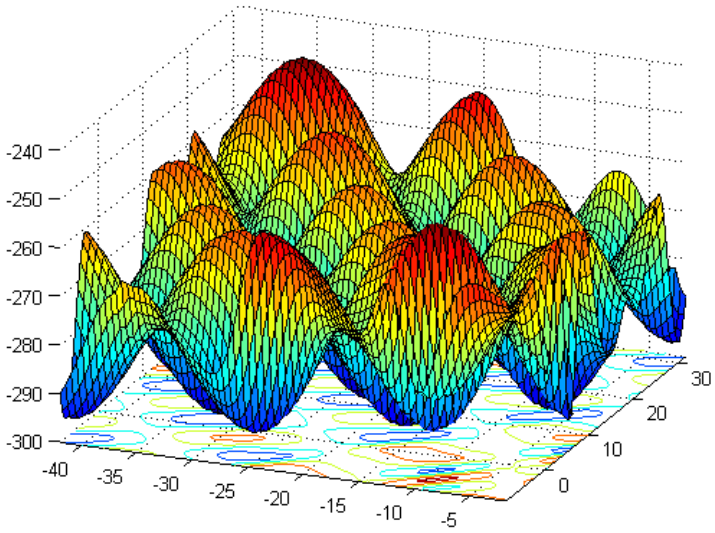

Figure 14. Rotated Rastrigin's Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Local optima's number is huge

4.11. Non-Continuous Rotated Rastrigin's Function (F11)

$$f_{11}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{11}^*$$

$$\hat{x} = M_1 \frac{5.12(x-o)}{100}, y_i = \begin{cases} \hat{x}_i & if\ |\hat{x}_i| \leq 0.5 \\ \frac{round(2\hat{x}_i)}{2} & if|\hat{x}_i| > 0.5 \end{cases} for\ i = 1,2,\dots D$$

$$z = M_1\Lambda^{10}M_2T_{asy}^{0.2}(T_{osz}(y))$$

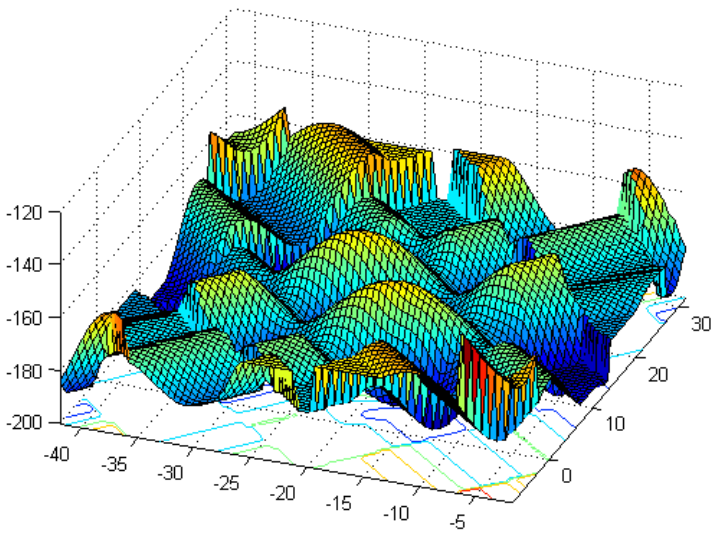

Figure 15. Non-continuous Rotated Rastrigin's Function **[15]**

Properties:

- Multi-modal
- Rotated
- Non-separable
- Asymmetrical
- Local optima's number is large
- Non-continuous

4.12. Schwefel's Function (F12)

$$f_{12}(z) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i) + f_{12}^{*}$$

$$z = \Lambda^{10}\left(\frac{1000(x-o)}{100}\right) + 4.209687462275036e+002$$

$$g(z_i) = \begin{cases} z_i \sin\left(|z_i|^{\frac{1}{2}}\right) & if\ |z_i| \le 500 \\ \left(500 - mod(z_i, 500)\right)\sin\left(\sqrt{\left|500 - mod\left(z_{i,}, 500\right)\right|}\right) - \dfrac{(z_i - 500)^2}{1000D} & if\ z_i > 500 \\ \left(mod(|z_i|, 500) - 500\right)\sin\left(\sqrt{|mod(|z_i|, 500) - 500|}\right) - \dfrac{(z_i + 500)^2}{1000D} & if\ z_i < -500 \end{cases}$$

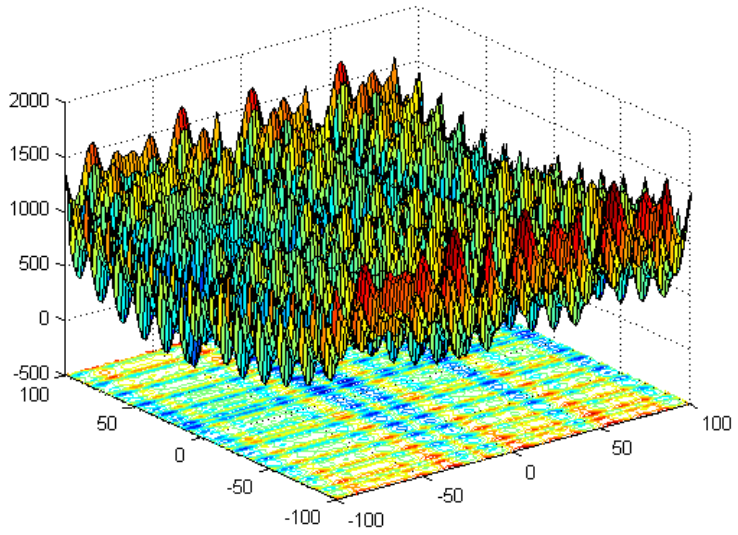

Figure 16. Schwefel's Function **[15]**

Properties:

- Multi-modal
- Rotated

- Non-separable
- Asymmetrical
- Local optima's number is huge and second better local optimum is far from the global optimum

4.13. Rotated Katsuura Function (F13)

$$f_{13}(x) = \frac{10}{D^2}\prod_{i=1}^{D}(1 + i\sum_{j=1}^{32}\frac{\left|2^j z_i - round(2^j z_i)\right|}{2^j})^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} + f_{13}^*$$

$$z = M_2 \Lambda^{100}(M_1 \frac{5(x-o)}{100})$$

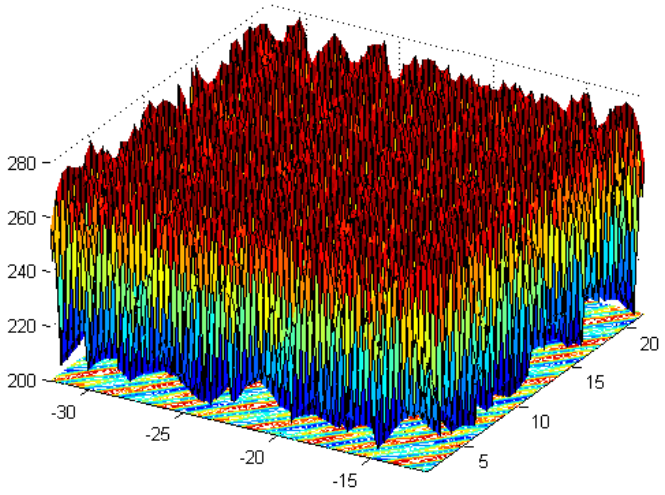

Figure 17. Rotated Katsuura Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Continuous everywhere yet differentiable nowhere

4.14. Lunacek Bi-Rastrigin Function (F14)

$$f_{14}(x) = \min\left(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, dD + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2\right) + 10\left(D - \sum_{i=1}^{D}\cos(2\pi\hat{z}_i)\right) + f_{14}^*$$

$$\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{{\mu_0}^2 - d}{s}}, s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, d = 1$$

$$y = \frac{10(x-o)}{100}, \hat{x}_i = 2sign(x_i^*)y_i + \mu_0, for\ i = 1,2,\dots,D$$

$$z = \Lambda^{100}(\hat{x} - \mu_0)$$

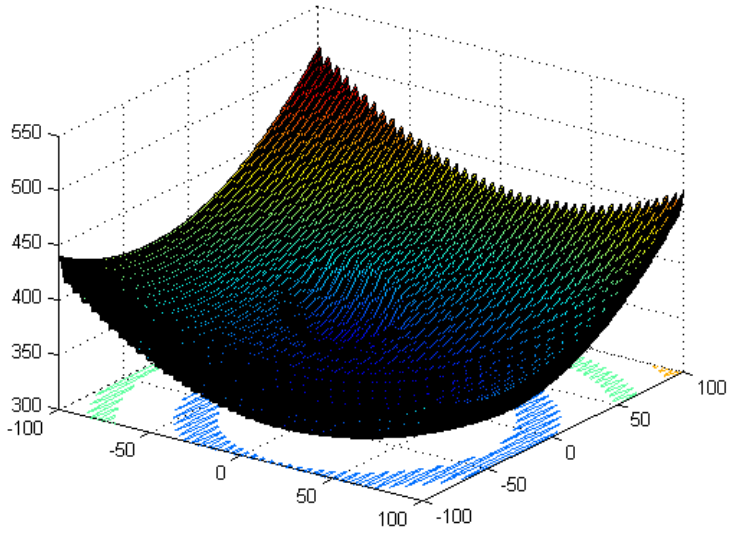

Figure 18. Lunacek bi-Rastrigin Function **[15]**

Properties:

- Multi-modal
- Asymmetrical

4.15. Rotated Lunacek Bi-Rastrigin Function (F15)

$$f_{15}(x) = \min\left(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, dD + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2\right) + 10\left(D - \sum_{i=1}^{D}\cos(2\pi\hat{z}_i)\right) + f_{14}^*$$

$$\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{{\mu_0}^2 - d}{s}}, s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, d = 1$$

$$y = \frac{10(x-o)}{100}, \hat{x}_i = 2sign(y_i^*)y_i + \mu_0, for\ i = 1,2,\dots,D$$

$$z = M_2\Lambda^{100}(M_1(\hat{x} - \mu_0))$$

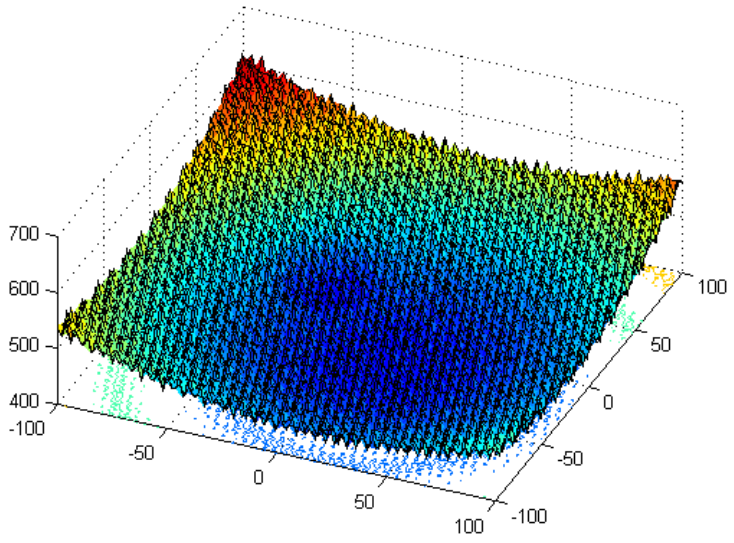

Figure 19. Rotated Lunacek Bi-Rastrigin Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Continuous everywhere yet differentiable nowhere

4.16. Rotated Expanded Griewank's plus Rosenbrock's Function (F16)

Basic Griewank's Function: $g_1(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$

Basic Rosenbrock's Function: $g_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$

$$f_{16}(x) = g_1(g_2(z_1, z_2)) + g_1(g_2(z_2, z_3)) + \cdots + g_1(g_2(z_{D-1}, z_D)) + g_1(g_2(z_D, z_1)) + f_{16}^*$$

$$z = M_1 \left(\frac{5(x - o)}{100} \right) + 1$$

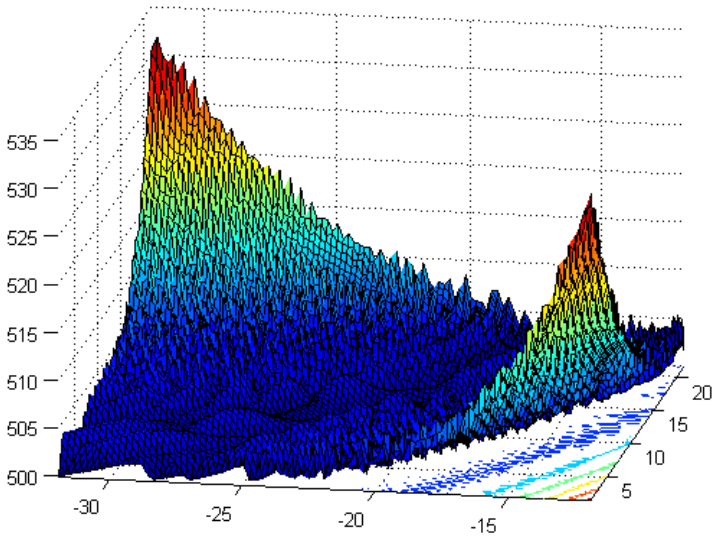

Figure 20. Rotated Expanded Griewank's plus Rosenbrock's Function **[15]**

Properties:

- Multi-modal
- Non-separable

4.17. Rotated Expanded Scaffer's Function (F17)

Scaffer's Function: $g(x,y) = 0.5 + \frac{(sin^2\left(\sqrt{x^2+y^2}\right)-0.5)}{(1+0.001(x^2+y^2))^2}$

$$f_{17}(x) = g(z_1, z_2) + g(z_2, z_3) + \cdots + g(z_{D-1}, z_D) + g(z_D, z_1) + f_{17}^*$$

$$z = M_2 T_{asy}^{0.5}(M_1(x - o))$$

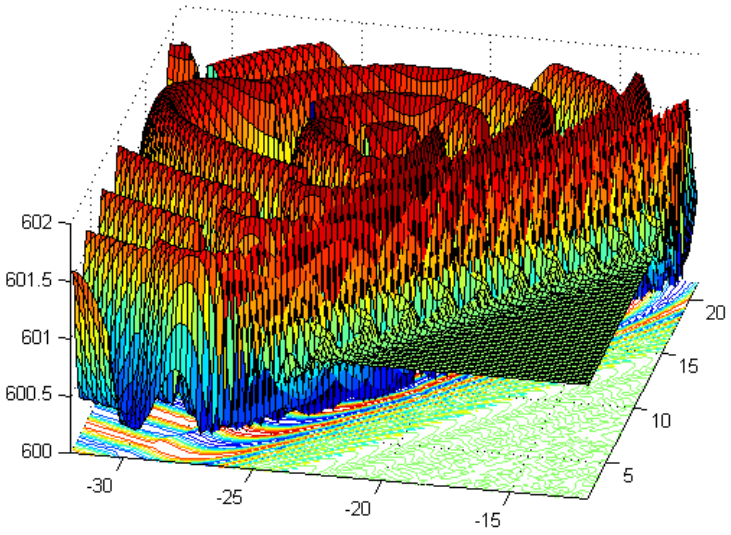

Figure 21. Rotated Expanded Scaffer's Function **[15]**

Properties:

- Multi-modal
- Non-separable
- Asymmetrical

5. EXPERIMENTAL SETUP AND RESULTS

5.1. Experimental Setup

As explained in the abstract, we applied variants of Particle swarm optimization on various benchmark functions in multiple dimensions, using the computational procedure to find the optimal solutions for those functions.

We used Java programming language and Eclipse IDE for writing the procedure. We have a Main program that has the following methods:

Method: setProperties

This method imports the properties specified in a property file, into the program. Variable parameters such as number of dimensions, number of runs that the program will go through, positive acceleration constants c_1 and c_2, benchmark function name that the PSO algorithm and its variants will be applied upon, and any other parameters that are needed for the variants of PSO.

Method: initializePopulation

This method initializes i number of particles with j number of dimensions that are specified in the properties file.

Method: initializeFitness

For the particles initialized in the above method, **initializeFitness** method calculates the fitness of the particles based on the function that is being used for the run of the program.

Method: initialVelocityValues

Similar to the **initializeFitness** function, **initialVelocityValues** method calculates velocity for all the initialized particles and all their dimensions.

Method: getAverage

Best fitness of all the fitness values calculated in each run is stored in an Array List. And, at the end of all the runs, the average is calculated when this method is called and returns the value.

Method: getVariance

This method calculates the variance of all the fitness values.

Method: getSD

This method calculates the standard deviation of all the fitness values.

We created Java classes for each of the variants of the PSO. The variants of the PSO that we ran the tests for are the following:

1. Basic Particle Swarm Optimization
2. Decreasing Weight Particle Swarm Optimization
3. Particle Swarm Optimization with Constriction
4. Time-Varying Particle Swarm Optimization
5. Time-Varying Inertia Weight Particle Swarm Optimization
6. Time-Varying Acceleration Coefficients Particle Swarm Optimization
7. Self-Organizing Hierarchical Particle Swarm Optimization with Time Varying - Acceleration Coefficients

We ran the above variants of the PSO algorithm, 51 times on each of the 17-benchmark functions and computed the average, variance and standard deviation for 10, 30, and 50 dimensions. Using the results, we found out suitable variations of the algorithm for the benchmark functions by considering the minimum optimal solution produced by each variant.

Following are the 17-benchmark functions that were used for the experiments:

F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17

5.1.1. Experimental Settings

Benchmark Functions/Problems: 17 Minimization Functions/Problems

Dimensions: 10, 30 and 50

Number of Runs per each Benchmark Function/Problem: 51

Number of Particles: $7 \times Dimension$

= 70 particles for 10 Dimensions

= 210 for particles 30 Dimensions

= 350 particles for 50 Dimensions

Number of Iterations: $(10000 \times \boldsymbol{Dimension}) / \boldsymbol{Number\ of\ Particles}$ = 1428

Search Range: $[-100, 100]^D$, D (Dimension) = 10, 30, and 50

Acceleration Constants: c_1 = c_2 = 2.0

Inertia: w = **0.6**

Initialization: Random initialization of particles within the search space.

5.2. Results

5.2.1. BPSO results on Problems with 10, 30, and 50 Dimensions

Below are the results of the runs that were run on 17 Benchmark functions (problems) using Basic Particle Swarm Optimization algorithm. Runs were made with the dimensions of 10, 30, and 50. As seen in Table 1, BPSO performed well for the Functions F1 and F3 on all the dimensions that were tested. With the Benchmark function F2, the algorithm fared well in dimensions 10 and 30. But, as the dimensions increased, the results were slightly increased above zero. Also in the table, you can see the variance and Standard deviation. For all other functions, we observed that as the dimensions increased, the values are higher indicating that performance of optimization decreases with the increase of dimensions.

Table 1. Basic PSO - Average, Variance and SD on 10, 30 and 50 Dimensions

Benchmark	Dimensions	Average	Variance	Standard Deviation
F1	10	**0.00E+00**	0.00E+00	0.00E+00
	30	**8.00E-11**	4.14E-20	2.04E-10
	50	**1.03E-03**	2.47E-06	1.57E-03
F2	10	**4.34E-03**	1.72E-04	1.31E-02
	30	**4.07E-03**	8.28E-04	2.88E-02
	50	3.21E+00	9.57E+01	9.78E+00
F3	10	**0.00E+00**	3.04E-27	5.51E-14
	30	**4.65E-06**	7.33E-11	8.56E-06
	50	**1.73E-01**	1.30E-02	1.14E-01
F4	10	6.15E+00	8.99E+00	3.00E+00
	30	3.15E+01	1.65E+02	1.28E+01
	50	1.24E+02	2.03E+03	4.50E+01
F5	10	1.43E+01	1.21E+02	1.10E+01
	30	1.67E+03	1.35E+06	1.16E+03
	50	1.23E+03	6.49E+05	8.05E+02
F6	10	2.07E+01	5.72E-02	2.39E-01
	30	2.10E+01	7.84E-03	8.85E-02
	50	2.12E+01	4.48E-03	6.69E-02
F7	10	9.75E+00	8.88E+00	2.98E+00
	30	3.77E+01	7.22E+01	8.50E+00
	50	7.50E+01	9.69E+01	9.84E+00
F8	10	3.76E-01	2.81E-02	1.68E-01
	30	1.69E+00	4.43E-01	6.65E-01
	50	3.77E+01	4.42E+02	2.10E+01
F9	10	2.69E+00	2.79E+00	1.67E+00
	30	2.65E+01	5.61E+01	7.49E+00
	50	7.10E+01	2.20E+02	1.48E+01
F10	10	1.61E+01	4.78E+01	6.92E+00
	30	9.57E+01	9.36E+02	3.06E+01
	50	2.37E+02	6.97E+03	8.35E+01
F11	10	2.23E+01	9.51E+01	9.75E+00
	30	1.75E+02	1.14E+03	3.38E+01
	50	4.41E+02	2.07E+03	4.55E+01
F12	10	1.97E+02	2.40E+04	1.55E+02
	30	1.45E+03	9.84E+04	3.14E+02
	50	2.73E+03	2.77E+05	5.27E+02

Table 1. Basic PSO - Average, Variance and SD on 10, 30 and 50 Dimensions (continued)

Benchmark	Dimensions	Average	Variance	Standard Deviation
F13	10	**8.32E-01**	1.15E-01	3.39E-01
	30	1.69E+00	8.59E-01	9.27E-01
	50	3.34E+00	9.39E-02	3.06E-01
F14	10	1.31E+01	1.86E+01	4.31E+00
	30	7.62E+01	1.23E+02	1.11E+01
	50	1.89E+02	1.14E+03	3.38E+01
F15	10	2.95E+01	7.84E+01	8.85E+00
	30	2.37E+02	9.22E+02	3.04E+01
	50	5.30E+02	1.18E+03	3.44E+01
F16	10	**8.28E-01**	8.77E-02	2.96E-01
	30	7.94E+00	6.95E+00	2.64E+00
	50	1.60E+01	2.73E+01	5.23E+00
F17	10	4.28E+00	1.55E-01	3.94E-01
	30	1.37E+01	6.12E-02	2.47E-01
	50	2.42E+01	1.43E-01	3.78E-01

5.2.2. DWPSO results on Problems with 10, 30, and 50 Dimensions

Below are the results of the runs that were run on 17 Benchmark functions (problems) using Decreasing weight Particle Swarm Optimization. Runs were made with the dimensions of 10, 30, and 50. As seen in Table 2, Decreasing weight Particle Swarm optimization has also fared well with the benchmark functions F1, F2, and F3 in all three dimensions 10, 30, and 50. All the values are 0 or less with good optimizations. With the Benchmark function F8, F13 and F16, the algorithm fared well in dimensions 10. Although DWPSO is expected to reveal better results compared to PSO because PSO is an improved variation of Basic PSO, we noticed that the results are not very different from BPSO.

Table 2. DWPSO - Average, Variance and SD on 10, 30 and 50 Dimensions

Benchmark	Dimensions	Average	Variance	Standard Deviation
F1	10	**0.00E+00**	0.00E+00	0.00E+00
	30	**1.04E-06**	1.83E-11	4.28E-06
	50	**4.36E-02**	9.32E-03	9.66E-02
F2	10	**0.00E+00**	0.00E+00	0.00E+00
	30	**6.66E-07**	7.62E-12	2.76E-06
	50	**3.17E-02**	8.89E-03	9.43E-02
F3	10	**0.00E+00**	5.58E-27	7.47E-14
	30	**1.79E-04**	2.08E-07	4.56E-04
	50	**2.60E-01**	3.92E-02	1.98E-01
F4	10	6.26E+00	3.73E+00	1.93E+00
	30	3.59E+01	4.41E+02	2.10E+01
	50	6.05E+01	7.32E+02	2.71E+01
F5	10	6.75E+00	5.21E+01	7.22E+00
	30	3.60E+02	7.86E+04	2.80E+02
	50	2.92E+03	2.15E+06	1.47E+03
F6	10	2.05E+01	9.39E-03	9.69E-02
	30	2.10E+01	8.27E-03	9.10E-02
	50	2.12E+01	6.18E-03	7.86E-02
F7	10	1.02E+01	5.84E+00	2.42E+00
	30	4.15E+01	2.63E+01	5.13E+00
	50	7.48E+01	5.79E+01	7.61E+00
F8	10	**4.11E-01**	5.00E-02	2.24E-01
	30	3.26E+00	4.82E+00	2.20E+00
	50	4.82E+01	8.77E+02	2.96E+01
F9	10	2.17E+00	1.78E+00	1.33E+00
	30	2.96E+01	7.10E+01	8.43E+00
	50	8.78E+01	2.40E+02	1.55E+01
F10	10	1.62E+01	4.96E+01	7.04E+00
	30	9.04E+01	5.52E+02	2.35E+01
	50	2.28E+02	3.32E+03	5.76E+01
F11	10	2.42E+01	5.56E+01	7.46E+00
	30	1.71E+02	1.54E+03	3.92E+01
	50	4.14E+02	3.26E+03	5.71E+01
F12	10	2.21E+02	3.18E+04	1.78E+02
	30	2.53E+03	4.07E+05	6.38E+02
	50	4.89E+03	6.13E+05	7.83E+02
	50	2.03E+00	1.83E+00	1.35E+00

Table 2. DWPSO - Average, Variance and SD on 10, 30 and 50 Dimensions (continued)

Benchmark	Dimensions	Average	Variance	Standard Deviation
F13	10	**4.92E-01**	2.20E-01	4.69E-01
	30	1.03E+00	1.06E+00	1.03E+00
	50	2.03E+00	1.83E+00	1.35E+00
F14	10	1.53E+01	6.00E+00	2.45E+00
	30	8.48E+01	3.42E+02	1.85E+01
	50	1.96E+02	9.40E+02	3.07E+01
F15	10	3.15E+01	6.79E+01	8.24E+00
	30	2.31E+02	7.19E+02	2.68E+01
	50	5.17E+02	3.63E+03	6.03E+01
F16	10	**9.74E-01**	1.15E-01	3.39E-01
	30	6.32E+00	6.21E+00	2.49E+00
	50	1.57E+01	2.63E+01	5.13E+00
F17	10	3.87E+00	8.97E-02	2.99E-01
	30	1.35E+01	1.45E-01	3.81E-01
	50	2.44E+01	7.43E-02	2.73E-01

5.2.3. PSO_C results on Problems with 10, 30, and 50 Dimensions

Table 3 shows the results of the runs that were run on 17 Benchmark functions (problems) using Particle Swarm Optimization with Constriction Factor. Runs were made with the dimensions of 10, 30, and 50. As can be seen in the table below, PSO_C performed well for the functions F1, F2 and F3 on dimension of 10 that were tested. But, as the dimensions increased, the results were slightly increased above zero. With the Benchmark function F8 and F13 the algorithm fared well in dimension of 10. Also in the table, you can see the variance and Standard deviation. For all other functions, we observed that as the dimensions increased, the values are higher indicating that performance of optimization decreases with the increase of dimensions.

Table 3. PSO_C - Average, Variance and SD on 10, 30 and 50 Dimensions

Benchmark	Dimensions	Average	Variance	Standard Deviation
F1	10	**0.00E+00**	0.00E+00	0.00E+00
	30	8.68E+00	1.99E+02	1.41E+01
	50	5.09E+02	1.82E+05	4.27E+02
F2	10	**0.00E+00**	1.01E-27	3.18E-14
	30	4.18E+00	6.86E+01	8.28E+00
	50	4.96E+02	1.84E+05	4.29E+02
F3	10	**0.00E+00**	1.29E-26	1.14E-13
	30	8.41E+00	1.62E+02	1.27E+01
	50	1.14E+02	7.27E+03	8.53E+01
F4	10	6.77E+00	4.38E+00	2.09E+00
	30	4.90E+01	1.10E+03	3.32E+01
	50	1.57E+02	2.52E+03	5.02E+01
F5	10	1.24E+01	1.04E+02	1.02E+01
	30	1.07E+02	1.71E+03	4.14E+01
	50	6.58E+02	1.08E+05	3.29E+02
F6	10	2.05E+01	4.03E-03	6.35E-02
	30	2.10E+01	6.58E-03	8.11E-02
	50	2.11E+01	1.66E-03	4.08E-02
F7	10	8.11E+00	1.19E+01	3.45E+00
	30	4.00E+01	7.60E+01	8.72E+00
	50	7.25E+01	1.39E+02	1.18E+01
F8	10	**5.46E-01**	1.27E-01	3.56E-01
	30	3.46E+01	1.33E+03	3.65E+01
	50	2.50E+02	1.27E+04	1.13E+02
F9	10	6.07E+00	1.04E+01	3.22E+00
	30	6.82E+01	3.18E+02	1.78E+01
	50	1.95E+02	1.74E+03	4.17E+01
F10	10	1.81E+01	6.66E+01	8.16E+00
	30	1.17E+02	1.63E+03	4.04E+01
	50	2.45E+02	3.03E+03	5.50E+01
F11	10	2.32E+01	7.34E+01	8.57E+00
	30	1.90E+02	1.04E+03	3.23E+01
	50	4.13E+02	5.86E+03	7.66E+01
F12	10	2.85E+02	2.05E+04	1.43E+02
	30	2.07E+03	1.82E+05	4.27E+02
	50	4.32E+03	4.97E+05	7.05E+02

Table 3. PSO_C - Average, Variance and SD on 10, 30 and 50 Dimensions (continued)

Benchmark	Dimensions	Average	Variance	Standard Deviation
F13	10	**8.23E-01**	6.46E-02	2.54E-01
	30	2.10E+00	1.30E-01	3.60E-01
	50	2.88E+00	1.60E-01	4.00E-01
F14	10	1.51E+01	6.08E+00	2.47E+00
	30	1.00E+02	4.83E+02	2.20E+01
	50	2.80E+02	6.10E+03	7.81E+01
F15	10	2.35E+01	7.15E+01	8.45E+00
	30	1.43E+02	1.39E+03	3.73E+01
	50	3.58E+02	5.64E+03	7.51E+01
F16	10	8.93E-01	1.15E-01	3.40E-01
	30	7.64E+00	8.95E+00	2.99E+00
	50	5.85E+01	3.00E+03	5.48E+01
F17	10	4.38E+00	8.11E-02	2.85E-01
	30	1.40E+01	2.42E-01	4.92E-01
	50	2.33E+01	3.44E-01	5.87E-01

5.2.4. SHPSO_TVAC results on Problems with 10, 30, and 50 Dimensions

Table 4 shows the results of the runs that were run on 17 Benchmark functions (problems) using Self Organizing Hierarchical Particle Swarm Optimization with Time varying Acceleration Coefficients Algorithm. Runs were made with the dimensions of 10, 30, and 50. As seen in the table, SHPSO_TVAC performed well for the functions F1, F2 and F3 on dimension of 10 that were tested. But, as the dimensions increased, the results were slightly increased above zero. With the Benchmark function F8 and F13 the algorithm fared well in dimension of 10. Also in the table, you can see the variance and Standard deviation. For all other functions, we observed that as the dimensions increased, the values are higher indicating that performance of optimization decreases with the increase of dimensions.

Table 4. SHPSO_TVAC - Average, Variance and SD on 10, 30 and 50 Dimensions

Benchmark	Dimensions	Average	Variance	Standard Deviation
F1	10	**2.17E-07**	9.61E-13	9.80E-07
	30	4.26E+00	3.41E+01	5.84E+00
	50	3.54E+02	7.59E+04	2.76E+02
F2	10	**4.48E-08**	4.01E-14	2.00E-07
	30	3.80E+00	1.99E+01	4.46E+00
	50	1.91E+02	2.45E+04	1.56E+02
F3	10	**8.97E-06**	5.15E-10	2.27E-05
	30	5.73E+00	3.79E+01	6.15E+00
	50	1.73E+02	1.65E+04	1.28E+02
F4	10	7.60E+00	2.31E+00	1.52E+00
	30	3.98E+01	6.00E+02	2.45E+01
	50	1.23E+02	2.54E+03	5.04E+01
F5	10	3.11E+01	5.09E+02	2.26E+01
	30	5.47E+02	5.42E+04	2.33E+02
	50	9.42E+02	7.95E+04	2.82E+02
F6	10	2.05E+01	5.76E-03	7.59E-02
	30	2.10E+01	7.31E-03	8.55E-02
	50	2.11E+01	2.24E-03	4.73E-02
F7	10	6.49E+00	6.26E+00	2.50E+00
	30	3.59E+01	4.78E+01	6.91E+00
	50	5.48E+01	2.37E+01	4.87E+00
F8	10	1.52E+00	8.99E-01	9.48E-01
	30	7.39E+01	2.02E+03	4.50E+01
	50	5.98E+02	6.65E+04	2.58E+02
F9	10	**6.48E-01**	1.13E+00	1.06E+00
	30	4.90E+01	2.20E+02	1.48E+01
	50	1.61E+02	2.05E+03	4.53E+01
F10	10	2.84E+01	1.53E+02	1.24E+01
	30	2.35E+02	2.24E+03	4.73E+01
	50	5.43E+02	8.90E+03	9.43E+01
F11	10	2.82E+01	9.84E+01	9.92E+00
	30	2.26E+02	1.51E+03	3.89E+01
	50	5.14E+02	3.91E+03	6.25E+01
F12	10	1.76E+02	8.33E+03	9.12E+01
	30	1.66E+03	9.58E+04	3.09E+02
	50	4.37E+03	3.70E+05	6.09E+02

Table 4. SHPSO_TVAC - Average, Variance and SD on 10, 30 and 50 Dimensions (continued)

Benchmark	Dimensions	Average	Variance	Standard Deviation
F13	10	**7.77E-01**	5.67E-02	2.38E-01
	30	2.28E+00	1.34E-01	3.66E-01
	50	3.31E+00	6.52E-02	2.55E-01
F14	10	1.52E+01	6.70E+00	2.59E+00
	30	1.28E+02	9.10E+02	3.02E+01
	50	3.16E+02	4.77E+03	6.91E+01
F15	10	3.14E+01	4.41E+01	6.64E+00
	30	2.63E+02	2.97E+03	5.45E+01
	50	6.50E+02	1.17E+04	1.08E+02
F16	10	2.19E+00	9.22E-01	9.60E-01
	30	1.55E+01	2.84E+01	5.33E+00
	50	5.11E+01	1.39E+02	1.18E+01
F17	10	4.35E+00	3.73E-01	6.11E-01
	30	1.40E+01	3.42E-01	5.85E-01
	50	2.39E+01	2.99E-01	5.46E-01

5.2.5. TVPSO results on Problems with 10, 30, and 50 Dimensions

Table 5 shows the results of the runs that were run on 17 Benchmark functions (problems) using Time Varying Particle Swarm Optimization. Runs were made with the dimensions of 10, 30, and 50. As seen in the table below, TVPSO performed well for the functions F1, F2 and F3 on dimensions 10 and 20 that were tested. But, as the dimensions increased, the results were slightly increased above zero. With the Benchmark function F8, F13 and F16, the algorithm fared well in dimension of 10. Also, in the table, you can see the variance and Standard deviation. For all other functions, we observed that as the dimensions increased, the values are higher indicating that the performance of the optimization decreases with the increase of dimensions.

Table 5. TVPSO - Average, Variance and SD on 10, 30 and 50 Dimensions

Benchmark	Dimensions	Average	Variance	Standard Deviation
F1	10	**0.00E+00**	1.01E-27	3.18E-14
	30	**6.39E-02**	1.20E-01	3.47E-01
	50	7.09E+01	3.07E+04	1.75E+02
F2	10	**0.00E+00**	0.00E+00	0.00E+00
	30	**1.45E-02**	3.56E-03	5.96E-02
	50	3.88E+01	2.82E+03	5.31E+01
F3	10	**0.00E+00**	6.59E-27	8.12E-14
	30	**3.72E-02**	3.08E-02	1.76E-01
	50	2.39E+00	9.78E+01	9.89E+00
F4	10	6.08E+00	4.58E+00	2.14E+00
	30	3.95E+01	7.16E+02	2.68E+01
	50	1.08E+02	2.47E+03	4.97E+01
F5	10	1.03E+01	8.03E+01	8.96E+00
	30	6.73E+01	4.74E+02	2.18E+01
	50	2.24E+02	4.71E+03	6.86E+01
F6	10	2.06E+01	9.48E-03	9.74E-02
	30	2.11E+01	9.28E-03	9.63E-02
	50	2.12E+01	4.38E-03	6.62E-02
F7	10	1.03E+01	1.09E+01	3.30E+00
	30	4.24E+01	4.86E+01	6.97E+00
	50	7.51E+01	1.00E+02	1.00E+01
F8	10	**4.69E-01**	1.68E-01	4.10E-01
	30	1.91E+00	3.56E+00	1.89E+00
	50	2.34E+01	3.00E+02	1.73E+01
F9	10	1.62E+00	1.39E+00	1.18E+00
	30	2.44E+01	3.97E+01	6.30E+00
	50	7.65E+01	3.31E+02	1.82E+01
F10	10	1.10E+01	2.91E+01	5.40E+00
	30	7.03E+01	4.98E+02	2.23E+01
	50	1.76E+02	1.73E+03	4.15E+01
F11	10	1.81E+01	4.84E+01	6.96E+00
	30	1.62E+02	1.25E+03	3.53E+01
	50	3.18E+02	3.33E+03	5.77E+01
F12	10	1.48E+02	1.61E+04	1.27E+02
	30	1.16E+03	7.59E+04	2.76E+02
	50	3.26E+03	3.78E+05	6.15E+02

Table 5. TVPSO - Average, Variance and SD on 10, 30 and 50 Dimensions (continued)

Benchmark	Dimensions	Average	Variance	Standard Deviation
F13	10	**8.32E-01**	1.44E-01	3.79E-01
	30	1.35E+00	1.11E+00	1.05E+00
	50	1.71E+00	1.99E+00	1.41E+00
F14	10	1.31E+01	2.10E+00	1.45E+00
	30	6.52E+01	1.21E+02	1.10E+01
	50	1.56E+02	5.11E+02	2.26E+01
F15	10	2.39E+01	5.87E+01	7.66E+00
	30	1.44E+02	1.39E+03	3.73E+01
	50	3.16E+02	3.43E+03	5.85E+01
F16	10	**6.31E-01**	5.67E-02	2.38E-01
	30	5.58E+00	5.79E+00	2.41E+00
	50	2.65E+01	9.58E+01	9.79E+00
F17	10	4.32E+00	6.49E-01	8.05E-01
	30	1.42E+01	3.50E-01	5.91E-01
	50	2.41E+01	7.24E-01	8.51E-01

5.2.6. TVAC_PSO results on Problems with 10, 30, and 50 Dimensions

Table 6 shows the results of the runs that were run on 17 Benchmark functions (problems) using Time-Varying Acceleration Coefficients Particle Swarm Optimization. Runs were made with the dimensions of 10, 30, and 50. As seen in the table, TVAC_PSO performed well for the functions F1, F2 and F3 on dimension of 10 and 20 that were tested. But, as the dimensions increased, the results were slightly increased above zero. With the Benchmark function F8, F13 and F16 the algorithm fared well in dimension of 10. Also in the table, you can see the Variance and Standard deviation. For all other functions, we observed that as the dimensions increased, the values are higher indicating that the performance of the optimization decreases with the increase of dimensions.

Table 6. TVAC_PSO - Average, Variance and SD on 10, 30 and 50 Dimensions

Benchmark	Dimensions	Average	Variance	Standard Deviation
F1	10	**0.00E+00**	5.07E-27	7.12E-14
	30	**2.15E-04**	4.40E-07	6.63E-04
	50	**6.01E-01**	1.39E+00	1.18E+00
F2	10	**0.00E+00**	0.00E+00	0.00E+00
	30	**7.63E-05**	1.78E-08	1.33E-04
	50	**4.89E-01**	1.21E+00	1.10E+00
F3	10	**3.21E-10**	4.49E-18	2.12E-09
	30	**1.47E-02**	9.10E-04	3.02E-02
	50	3.17E+00	6.16E+01	7.85E+00
F4	10	5.43E+00	3.25E+00	1.80E+00
	30	3.02E+01	1.85E+02	1.36E+01
	50	6.49E+01	9.10E+02	3.02E+01
F5	10	6.71E+00	3.29E+01	5.74E+00
	30	9.18E+01	9.71E+02	3.12E+01
	50	1.63E+03	1.76E+06	1.33E+03
F6	10	2.07E+01	3.73E-02	1.93E-01
	30	2.10E+01	3.86E-03	6.22E-02
	50	2.12E+01	6.78E-03	8.23E-02
F7	10	1.09E+01	8.09E+00	2.84E+00
	30	4.40E+01	4.00E+01	6.32E+00
	50	7.90E+01	6.19E+01	7.87E+00
F8	10	**5.25E-01**	1.42E-01	3.77E-01
	30	3.73E+00	6.70E+00	2.59E+00
	50	2.79E+01	1.85E+02	1.36E+01
F9	10	2.70E+00	1.80E+00	1.34E+00
	30	3.42E+01	7.40E+01	8.61E+00
	50	1.04E+02	5.28E+02	2.30E+01
F10	10	1.42E+01	6.03E+01	7.77E+00
	30	8.30E+01	6.79E+02	2.61E+01
	50	1.97E+02	2.74E+03	5.24E+01
F11	10	2.06E+01	7.34E+01	8.57E+00
	30	1.54E+02	1.16E+03	3.41E+01
	50	3.47E+02	3.87E+03	6.22E+01
F12	10	2.58E+02	1.90E+04	1.38E+02
	30	2.46E+03	2.95E+05	5.43E+02
	50	5.60E+03	7.83E+05	8.85E+02

Table 6. TVAC_PSO - Average, Variance and SD on 10, 30 and 50 Dimensions (continued)

Benchmark	Dimensions	Average	Variance	Standard Deviation
F13	10	**5.17E-01**	2.01E-01	4.48E-01
	30	1.27E+00	1.11E+00	1.05E+00
	50	1.60E+00	2.24E+00	1.50E+00
F14	10	1.52E+01	8.61E+00	2.93E+00
	30	8.45E+01	1.36E+02	1.17E+01
	50	1.98E+02	1.20E+03	3.46E+01
F15	10	2.16E+01	2.97E+01	5.45E+00
	30	1.12E+02	5.34E+02	2.31E+01
	50	2.45E+02	2.31E+03	4.80E+01
F16	10	**7.58E-01**	6.91E-02	2.63E-01
	30	5.09E+00	4.63E+00	2.15E+00
	50	1.89E+01	5.35E+01	7.31E+00
F17	10	4.75E+00	2.93E-01	5.42E-01
	30	1.37E+01	3.15E-01	5.61E-01
	50	2.40E+01	2.86E-01	5.35E-01

5.2.7. TVIW_PSO results on Problems with 10, 30, and 50 Dimensions

Table 7 shows the results of the runs that were run on 17 Benchmark functions (problems) using Time-Varying Inertia Weight Particle Swarm Optimization. Runs were made with the dimensions of 10, 30, and 50. As seen in the table, TVIW_PSO performed well for the functions F1, F2 and F3 on dimension of 10 that were tested. But, as the dimensions increased, the results were slightly increased above zero. With the Benchmark function F13, the algorithm fared well in dimension of 10. Also in the table, you can see the Variance and Standard deviation. For all other functions, we observed that as the dimensions increased, the values are higher indicating that the performance of the optimization decreases with the increase of dimensions.

Table 7. TVIW_PSO - Average, Variance and SD on 10, 30 and 50 Dimensions

Benchmark	Dimensions	Average	Variance	Standard Deviation
F1	10	**4.52E-04**	6.86E-06	2.62E-03
	30	6.87E+02	7.34E+05	8.57E+02
	50	4.02E+03	3.29E+06	1.81E+03
F2	10	**1.78E-05**	1.59E-08	1.26E-04
	30	2.06E+02	4.55E+04	2.13E+02
	50	3.91E+03	3.55E+06	1.88E+03
F3	10	**9.97E-01**	2.39E+01	4.89E+00
	30	1.65E+02	5.22E+03	7.22E+01
	50	9.20E+02	2.72E+05	5.22E+02
F4	10	5.57E+00	4.30E+00	2.07E+00
	30	1.14E+02	2.73E+03	5.22E+01
	50	4.01E+02	1.93E+04	1.39E+02
F5	10	2.32E+01	1.91E+02	1.38E+01
	30	2.15E+02	9.14E+03	9.56E+01
	50	7.59E+02	1.99E+05	4.46E+02
F6	10	2.05E+01	2.75E-03	5.25E-02
	30	2.09E+01	3.54E-03	5.95E-02
	50	2.11E+01	1.94E-03	4.40E-02
F7	10	3.28E+00	3.33E+00	1.82E+00
	30	3.98E+01	5.64E+01	7.51E+00
	50	6.42E+01	1.17E+02	1.08E+01
F8	10	1.05E+00	1.97E+00	1.40E+00
	30	1.79E+02	1.11E+04	1.05E+02
	50	7.51E+02	6.56E+04	2.56E+02
F9	10	1.18E+01	2.64E+01	5.14E+00
	30	1.17E+02	1.11E+03	3.32E+01
	50	2.99E+02	2.84E+03	5.33E+01
F10	10	2.01E+01	8.64E+01	9.30E+00
	30	1.22E+02	1.24E+03	3.52E+01
	50	2.93E+02	4.37E+03	6.61E+01
F11	10	3.69E+01	2.33E+02	1.53E+01
	30	2.45E+02	3.62E+03	6.02E+01
	50	4.99E+02	9.53E+03	9.76E+01
F12	10	3.19E+02	1.35E+04	1.16E+02
	30	2.57E+03	2.86E+05	5.35E+02
	50	5.27E+03	5.06E+05	7.12E+02

Table 7. TVIW_PSO - Average, Variance and SD on 10, 30 and 50 Dimensions (continued)

Benchmark	Dimensions	Average	Variance	Standard Deviation
F13	10	**8.93E-01**	9.48E-02	3.08E-01
	30	1.63E+00	1.83E-01	4.27E-01
	50	2.19E+00	3.27E-01	5.72E-01
F14	10	1.90E+01	1.56E+01	3.95E+00
	30	1.51E+02	2.21E+03	4.70E+01
	50	4.77E+02	9.64E+03	9.82E+01
F15	10	2.52E+01	6.07E+01	7.79E+00
	30	1.83E+02	2.85E+03	5.34E+01
	50	5.26E+02	1.13E+04	1.06E+02
F16	10	1.04E+00	2.22E-01	4.71E-01
	30	2.47E+01	6.38E+02	2.53E+01
	50	6.10E+02	5.91E+06	2.43E+03
F17	10	4.46E+00	2.62E-01	5.12E-01
	30	1.43E+01	3.26E-01	5.71E-01
	50	2.37E+01	5.59E-01	7.47E-01

Presented in Figures 20-22 are the graphs for a few of the functions displaying the fitness versus number of iterations on different variants of the PSO algorithm.

Figure 20 displays the graph of the Sphere Function on the Basic PSO variant with number of iterations on the horizontal axis and the fitness values at each iteration on the vertical axis for the dimension of 10.

Figure 21 displays the graph of Different Power's Function on TVPSO variant with number of iterations on the horizontal axis and the fitness values at each iteration on the vertical axis for the dimension of 10.

Figure 22 displays the graph of Rastrigin's Function on SHPSO_TVAC variant with number of iterations on the horizontal axis and the fitness values at each iteration on the vertical axis for the dimension of 10.

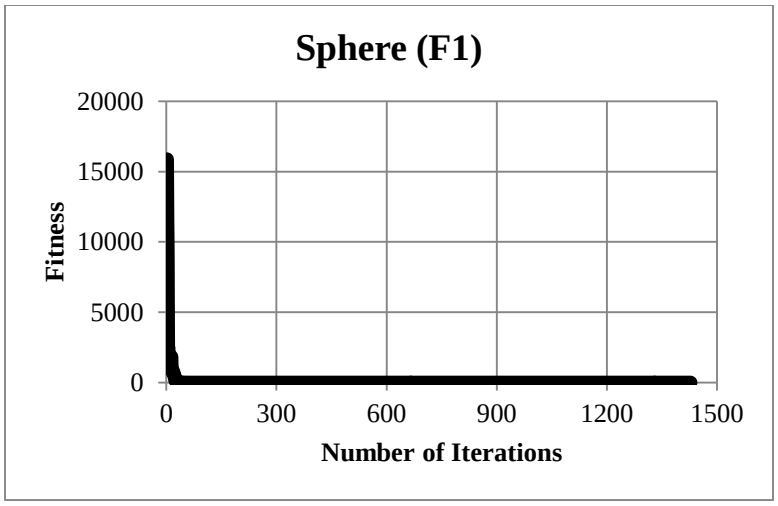

Figure 22. BPSO Variation of Fitness vs. Number of Iterations for F1 on 10D

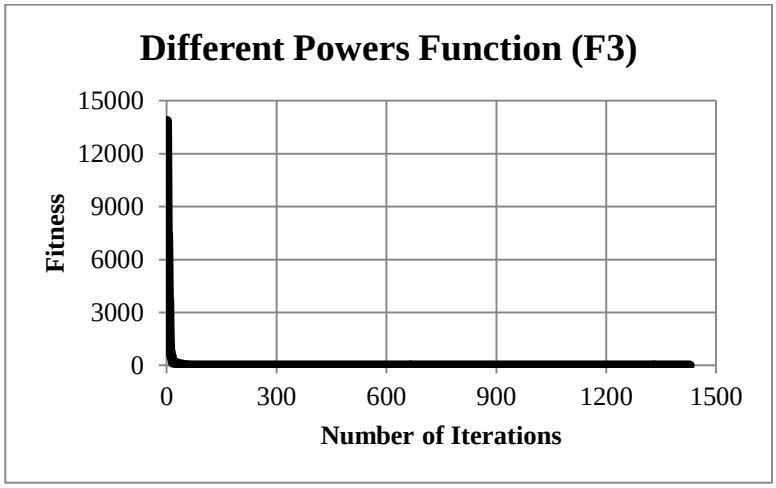

Figure 23. TVPSO Variation of Fitness vs. Number of Iterations for F3 on 10D

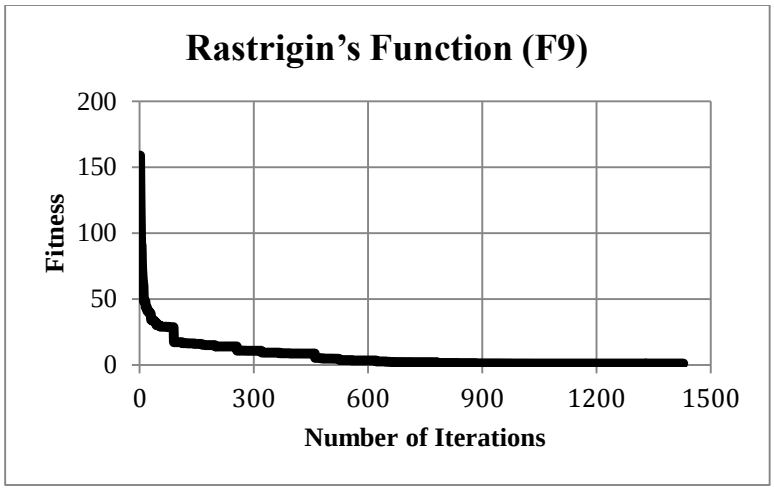

Figure 24. SHPSO_TVAC Variation of Fitness vs. Number of Iterations for F9 on 10D

5.3. Comparisons

Table 8 shows the results of all the variants of Particle Swarm Optimization that were run on 17 problems (Benchmark Functions). Runs were made with the dimensions of 10 and noted in the table. As can be seen, the results of all the variants for the functions F1, F2 and F3 fared well in dimension of 10. Among the 7 variants of PSO, TVPSO results are more consistent and optimal than other variants. PSO_C and SHPSO_TVAC are least consistent variants when compared to other variants of the algorithm.

Table 8. Comparisons of variants of PSO (dimensions of 10) on different problems

Benchmark	BPSO	DWPSO	PSO_C	SHPSO TVAC	TVPSO	TVAC PSO	TVIW PSO
F1	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.17E-07	**0.00E+00**	**0.00E+00**	4.52E-04
F2	4.34E-03	**0.00E+00**	**0.00E+00**	4.48E-08	**0.00E+00**	**0.00E+00**	1.78E-05
F3	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.97E-06	**0.00E+00**	3.21E-10	9.97E-01
F4	6.15E+00	6.26E+00	6.77E+00	7.60E+00	6.08E+00	**5.43E+00**	5.57E+00
F5	1.43E+01	6.75E+00	1.24E+01	3.11E+01	1.03E+01	**6.71E+00**	2.32E+01

Table 8. Comparisons of variants of PSO (dimensions of 10) on different problems (continued)

Benchmark	BPSO	DWPSO	PSO_C	SHPSO TVAC	TVPSO	TVAC PSO	TVIW PSO
F6	2.07E+01	**2.05E+01**	**2.05E+01**	**2.05E+01**	2.06E+01	2.07E+01	**2.05E+01**
F7	9.75E+00	1.02E+01	8.11E+00	6.49E+00	1.03E+01	1.09E+01	**3.28E+00**
F8	**3.76E-01**	4.11E-01	5.46E-01	1.52E+00	4.69E-01	5.25E-01	1.05E+00
F9	2.69E+00	2.17E+00	6.07E+00	**6.48E-01**	1.62E+00	2.70E+00	1.18E+01
F10	1.61E+01	1.62E+01	1.81E+01	2.84E+01	**1.10E+01**	1.42E+01	2.01E+01
F11	2.23E+01	2.42E+01	2.32E+01	2.82E+01	**1.81E+01**	2.06E+01	3.69E+01
F12	1.97E+02	2.21E+02	2.85E+02	1.76E+02	**1.48E+02**	2.58E+02	3.19E+02
F13	8.32E-01	**4.92E-01**	8.23E-01	7.77E-01	8.32E-01	5.17E-01	8.93E-01
F14	1.31E+01	1.53E+01	1.51E+01	1.52E+01	**1.31E+01**	1.52E+01	1.90E+01
F15	2.95E+01	3.15E+01	2.35E+01	3.14E+01	2.39E+01	**2.16E+01**	2.52E+01
F16	8.28E-01	9.74E-01	8.93E-01	2.19E+00	**6.31E-01**	7.58E-01	1.04E+00
F17	4.28E+00	**3.87E+00**	4.38E+00	4.35E+00	4.32E+00	4.75E+00	4.46E+00

Table 9 shows the results of all the variants of Particle Swarm Optimization that were run on 17 problems (Benchmark Functions). Runs were made with the dimensions of 30 and noted in the table. As can be seen, the results of variants - BPSO, DWPSO, TVPSO, TVAC_PSO for the functions F1, F2 and F3 fared well in dimension of 20. PSO_C, SHPSO_TVAC results were near to the optimal solutions. TVIW_PSO variants results are not consistent as the dimensions are increased. Among the 7 variants of PSO, again TVPSO results are more consistent in dimensions 30 and optimal than other variants. PSO_C, SHPSO_TVAC and TVIW_PSO are least consistent variants when compared to other variants of the algorithm.

Table 9. Comparisons of variants of PSO (dimensions of 30) on different problems

Benchmark	BPSO	DWPSO	PSO_C	SHPSO TVAC	TVPSO	TVAC PSO	TVIW PSO
F1	**8.00E-11**	**1.04E-06**	8.68E+00	4.26E+00	**6.39E-02**	**2.15E-04**	6.87E+02
F2	**4.07E-03**	**6.66E-07**	4.18E+00	3.80E+00	**1.45E-02**	**7.63E-05**	2.06E+02
F3	**4.65E-06**	**1.79E-04**	8.41E+00	5.73E+00	**3.72E-02**	**1.47E-02**	1.65E+02
F4	3.15E+01	3.59E+01	4.90E+01	3.98E+01	3.95E+01	**3.02E+01**	1.14E+02
F5	1.67E+03	3.60E+02	1.07E+02	5.47E+02	**6.73E+01**	9.18E+01	2.15E+02
F6	2.10E+01	2.10E+01	2.10E+01	2.10E+01	2.11E+01	2.10E+01	**2.09E+01**
F7	3.77E+01	4.15E+01	4.00E+01	**3.59E+01**	4.24E+01	4.40E+01	3.98E+01
F8	**1.69E+00**	3.26E+00	3.46E+01	7.39E+01	1.91E+00	3.73E+00	1.79E+02
F9	2.65E+01	2.96E+01	6.82E+01	4.90E+01	**2.44E+01**	3.42E+01	1.17E+02
F10	9.57E+01	9.04E+01	1.17E+02	2.35E+02	**7.03E+01**	8.30E+01	1.22E+02
F11	1.75E+02	1.71E+02	1.90E+02	2.26E+02	1.62E+02	**1.54E+02**	2.45E+02
F12	1.45E+03	2.53E+03	2.07E+03	1.66E+03	**1.16E+03**	2.46E+03	2.57E+03
F13	1.69E+00	**1.03E+00**	2.10E+00	2.28E+00	1.35E+00	1.27E+00	1.63E+00
F14	7.62E+01	8.48E+01	1.00E+02	1.28E+02	**6.52E+01**	8.45E+01	1.51E+02
F15	2.37E+02	2.31E+02	1.43E+02	2.63E+02	1.44E+02	**1.12E+02**	1.83E+02
F16	7.94E+00	6.32E+00	7.64E+00	1.55E+01	5.58E+00	**5.09E+00**	2.47E+01
F17	1.37E+01	**1.35E+01**	1.40E+01	1.40E+01	1.42E+01	1.37E+01	1.43E+01

Table 10 shows the results of all the variants of Particle Swarm Optimization that were run on 17 problems (Benchmark Functions). Runs were made with the dimensions of 50 and noted in the table. As can be seen, the results of DWPSO variant for the functions F1, F2 and F3 fared well

in dimension of 10. BPSO algorithm fared well for the functions F1 and F3 whereas the TVAC_PSO fared well for functions F1 and F2. But, BPSO, DWPSO and TVAC_PSO algorithm results were near to the optimal solutions for the function F1, F2 and F3. Among the 7 variants of PSO, TVPSO results are more consistent and optimal than other variants. PSO_C, SHPSO_TVAC and TVIW_PSO are least consistent variants when compared to other variants of the algorithm. As the dimensions were increased, the results were not as consistent as the results were for smaller dimensions.

Table 10. Comparisons of variants of PSO (dimensions of 50) on different problems

Benchmark	BPSO	DWPSO	PSO_C	SHPSO TVAC	TVPSO	TVAC PSO	TVIW PSO
F1	**1.03E-03**	**4.36E-02**	5.09E+02	3.54E+02	7.09E+01	**6.01E-01**	4.02E+03
F2	3.21E+00	**3.17E-02**	4.96E+02	1.91E+02	3.88E+01	**4.89E-01**	3.91E+03
F3	**1.73E-01**	**2.60E-01**	1.14E+02	1.73E+02	2.39E+00	3.17E+00	9.20E+02
F4	1.24E+02	**6.05E+01**	1.57E+02	1.23E+02	1.08E+02	6.49E+01	4.01E+02
F5	1.23E+03	2.92E+03	6.58E+02	9.42E+02	**2.24E+02**	1.63E+03	7.59E+02
F6	2.12E+01	2.12E+01	**2.11E+01**	**2.11E+01**	2.12E+01	2.12E+01	**2.11E+01**
F7	7.50E+01	7.48E+01	7.25E+01	**5.48E+01**	7.51E+01	7.90E+01	6.42E+01
F8	3.77E+01	4.82E+01	2.50E+02	5.98E+02	**2.34E+01**	2.79E+01	7.51E+02
F9	**7.10E+01**	8.78E+01	1.95E+02	1.61E+02	7.65E+01	1.04E+02	2.99E+02
F10	2.37E+02	2.28E+02	2.45E+02	5.43E+02	**1.76E+02**	1.97E+02	2.93E+02
F11	4.41E+02	4.14E+02	4.13E+02	5.14E+02	**3.18E+02**	3.47E+02	4.99E+02
F12	**2.73E+03**	4.89E+03	4.32E+03	4.37E+03	3.26E+03	5.60E+03	5.27E+03
F13	3.34E+00	2.03E+00	2.88E+00	3.31E+00	1.71E+00	**1.60E+00**	2.19E+00

Table 10. Comparisons of variants of PSO (dimensions of 50) on different problems (continued)

Benchmark	BPSO	DWPSO	PSO_C	SHPSO TVAC	TVPSO	TVAC PSO	TVIW PSO
F14	1.89E+02	1.96E+02	2.80E+02	3.16E+02	**1.56E+02**	1.98E+02	4.77E+02
F15	5.30E+02	5.17E+02	3.58E+02	6.50E+02	3.16E+02	**2.45E+02**	5.26E+02
F16	1.60E+01	**1.57E+01**	5.85E+01	5.11E+01	2.65E+01	1.89E+01	6.10E+02
F17	2.42E+01	2.44E+01	**2.33E+01**	2.39E+01	2.41E+01	2.40E+01	2.37E+01

6. CONCLUSION AND FUTURE WORK

In this paper, we discussed the PSO algorithm and its variants, benchmark functions, the computational procedure we developed for running the PSO variants with 17 benchmark functions, and the results obtained from the runs. We ran the variants of the algorithm 51 times on each of the 17-benchmark functions and computed the average, variance and standard deviation for 10, 30, and 50 dimensions.

Based on the results of the experiments, we found the suitable variants of the algorithm for the benchmark functions by considering the minimum optimal solution produced by each variant. On comparing the results, we found that DWPSO outperforms other variants of PSO in higher dimensions for the functions such as Rotated High Conditioned Elliptic Function. For the Rotated Schaffer's Function (F5), TVPSO performance is better in higher dimensions than any other variants. Also, Time Varying PSO is more consistent on 10, 30 and 50 dimensions on most of the functions such as Rastrigin Functions, Schwefel's Functions. Variants of PSO such as DWPSO, TVAC_PSO, TVIW_PSO, BPSO also resulted in optimal solutions to few of the problems that were run. Results of SHPSO and PSO_Constriction variants are not as consistent as TVPSO variant.

The results indicate that Time Varying Particle Swarm Optimization is more consistent compared to other variants, and thus the future work of this paper can include different combinations on Time varying particle swarm optimization in order to find the optimal solutions on various dimensions.

Furthermore, more complex problems (benchmark functions) can be used to test the PSO variants. There are more variations of Particle swarm optimization algorithm such as CLPSO that can be tested with the benchmark functions. Also, we noticed that with the increase of dimensions,

convergence is less compared to the lower dimensions. So, instead of running the program with the same number of iterations for all dimensions, the iterations could be increased with the number of dimensions. So, for example, instead of having 1428 iterations for 10, 30, 50 dimensions, the algorithms could be ran with more than 1428 (may be double) iterations for 20 dimensions, and with more than 1428 (may be 5 times) iterations for 50 dimensions, to find where convergence occurs.

7. REFERENCES

[1] A. P. Engelbrecht, Computational Intelligence: An Introduction, Chichester, England: John Wiley & Sons, 2007.

[2] Y.-f. Zhu and X.-m. Tang, "Overview of Swarm Intelligence," *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, pp. V9-400 - V9-403, 2010.

[3] E. Bonabeau, M. Dorigo and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, New york, 1999.

[4] L. M. Plà-Aragonés, Ed., Handbook of Operations Research in Agriculture and the Agri-Food Industry, Lleida: Springer, 2015.

[5] D. H. Chadwick, "Weaver Ants," May 2011. [Online]. Available: http://ngm.nationalgeographic.com/2011/05/weaver-ants/chadwick-text. [Accessed 6 October 2015].

[6] H. Hosseini-Nasab and L. Emami, "A hybrid particle swarm optimisation for dynamic facility layout problem," *International Journal of Production Research*, p. 4325–4335, 2013.

[7] "The Problem," [Online]. Available: http://www.math.uwaterloo.ca/tsp/problem/index.html.

[8] M. Dorigo and Luca Maria Gambardella, "Ant colonies for the traveling salesman problem".

[9] C.-H. Yang, C.-J. Hsiao and L.-Y. Chuang, "Linearly Decreasing Weight Particle Swarm Optimization with Accelerated Strategy for Data Clustering," *IAENG International Journal of Computer Science,* 2010.

[10] R. C. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources".

[11] J. Polprasert and W. Ongsakul, "Chaotic based PSO with Time-Varying Acceleration Coefficients for Security Constrained Optimal Power Flow Problem," *International Conference and Utility Exhibition 2014 on Green Energy for Sustainable Development,* 2014.

[12] A. Sengupta and V. K. Mishra, "Time Varying vs. Fixed Acceleration Coefficient PSO Driven Exploration during High Level Synthesis: Performance and Quality Assessment".

[13] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,* 2004.

[14] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. -P. Chen, A. Auger and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," May 2005.

[15] J. J. Liang, B. Y. Qu, P. N. Suganthan and A. G. Hernández-Díaz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization," January 2013.